16. Конструкции за прекъсване

Проф. д-р Емил Хаджиколев

1. Прекъсване на изпълнението
2. Конструкция continue
3. Конструкция break
4. Конструкция return
5. Функция exit(0);

Прекъсване на изпълнението

- За прекъсване на изпълнението на основната работна логика на цикъл, конструкция за избор на вариант, подпрограма и програма се използват различни възможности:
  - конструкция break;
  - конструкция continue;
  - конструкция return;
  - функция exit()
- Обикновено, прекъсване е необходимо при възникване на някакво специално условие, проверявано в if конструкция или ако трябва да се опростят няколко сложни if-else конструкции.

Конструкция continue

- Конструкция continue **прекъсва текущата итерация** на цикъл.
- Ако в тялото на цикъла има конструкции, които следват извикването на continue, то те не се изпълняват.
- Изпълнението на цикъла продължава със следващата итерация.
- **По този начин, с една начална проверка в тялото на цикъл, може да си спестим else част като оставим по-изчистен и по-лесно четим код.**

Пример за continue – брой на отрицателните числа в масив (1)

```
// Връща броя на отрицателните числа в масив arr с размер size.
// Ако елементът е положителен, му се задава нова стойност = старата*2
int countNegative(int arr[], int size){
    int counter = 0; // брояч за отрицателните числа
    for (int i = 0; i < size; i++) {
        if (arr[i] < 0) {  // ако числото е отрицателно...
            counter++;     // ...увеличаваме брояча и...
            continue;      // ...прекъсваме текущата итерация
        }

        // това е "else"-частта на горния if, но без else
        arr[i] = arr[i] * 2;
    }
    return counter;
}
```

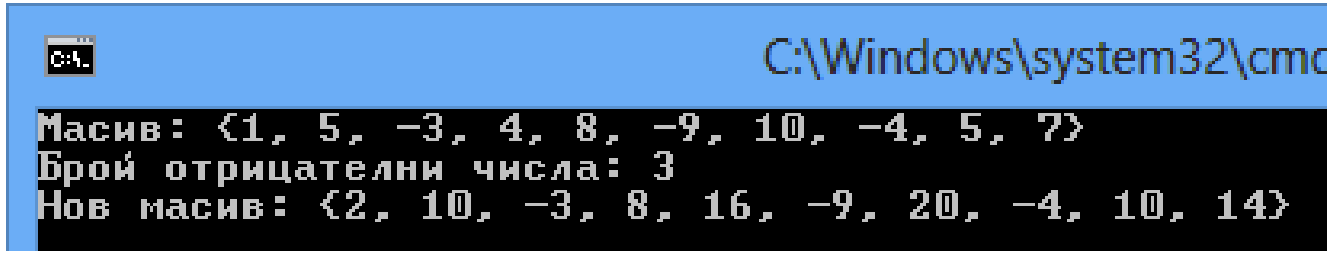
Пример за continue – брой на отрицателните числа в масив (2)

```
int main() {
    setlocale(LC_ALL, "bg");

    const int size = 10;
    int arr[size] = { 1, 5, -3, 4, 8, -9, 10, -4, 5, 7 };

    cout << "Масив: " << arrayToString(arr, size) << '\n'; // от предишната лекция
    cout << "Брой отрицателни числа: " << countNegative(arr, size) << '\n';
    cout << "Нов масив: " << arrayToString(arr, size) << "\n\n";

    return 0;
}
```



Конструкция break

- Конструкция break се използва за **прекъсване на цикъл и switch-case конструкции**.
- След извикването му се излиза от текущата конструкция (в която се ползва) и програмата продължава да се изпълнява със следващата конструкция.

Пример за break – търсене в масив на елемент, който се дели без остатък на дадено число (1)

```
// Връща първото число от масив arr (с размер size),
// което се дели на divisor без остатък.
// Ако не е намерено такова число се връща 0.
int searchNum(int arr[], int size, int divisor){
    int found = 0; // намерено число
    for (int i = 0; i < size; i++) {
        if (arr[i] % divisor==0) {  // ако остатъка при деление е 0...
            found =arr[i];          // ...записваме намереното число във found и...
            break;                  // ...търсенето и цикъла приключват
        }
    }

    return found;  // намереното число се връща като резултат
}
```

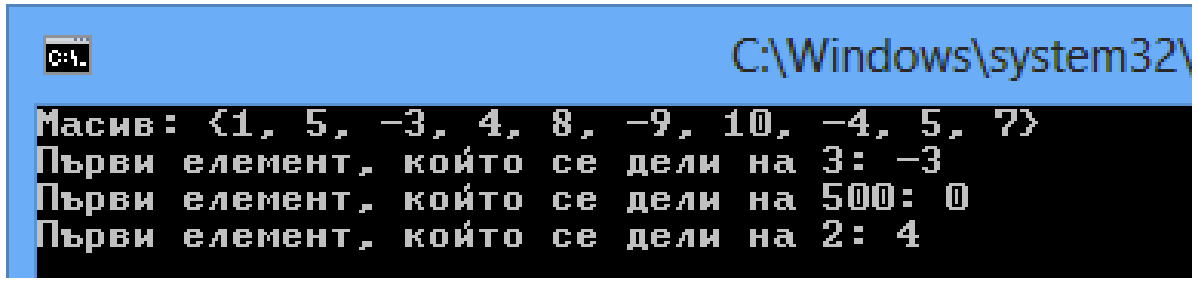
Пример за break – търсене в масив на елемент, който се дели без остатък на дадено число (2)

```
int main() {
    setlocale(LC_ALL, "bg");

    const int size = 10;
    int arr[size] = { 1, 5, -3, 4, 8, -9, 10, -4, 5, 7 };

    cout << "Масив: " << arrayToString(arr, size) << '\n';  // от предишната лекция
    cout << "Първи елемент, който се дели на 3: " << searchNum(arr, size, 3) << '\n';
    cout << "Първи елемент, който се дели на 500: " << searchNum(arr, size, 500) << '\n';
    cout << "Първи елемент, който се дели на 2: " << searchNum(arr, size, 2) << "\n\n";

    return 0;
}
```



Конструкция return

- Конструкция return прекъсва изпълнението на текущата **функция, но в частност и на всяка друга конструкция**.
- След изпълнението му, функцията приключва изпълнението си, и програмата продължава да се изпълнява от мястото на извикване на функцията.
- Ако функцията връща стойност, след return трябва да има израз или стойност от съответния тип.

Пример за return – по-добра реализация на търсенето...

```
// Връща първото число от масив arr (с размер size),
// което се дели на divisor без остатък.
// Ако не е намерено такова число се връща 0.
int searchNum(int arr[], int size, int divisor){
    for (int i = 0; i < size; i++) {
        if (arr[i] % divisor == 0) {  // ако остатъка при деление е 0...
            return arr[i];            // ...връщаме намереното число и...
                                      // ...функцията приключва
        }
    }
    return 0;  // ако не е намерено нищо в цикъла
}
```

Функция за прекъсване exit и return в main (1)

- С функция exit(<код за грешка>) да се прекъсне изпълнението на цялата програма.
- Като параметър се указва цяло число, което дава информация към операционната система (ОС) за грешката, аналогично на return в main-функцията:
  - 0 – приема се, че няма грешка
  - число, различно от 0 – грешка.
- ОС не е длъжна да обработва върнатия резултат.
- Смисълът на кодовете за изход се задават от програмиста и могат да се ползват от външни програми.

Функция за прекъсване exit и return в main (2)

- За разлика от return в main-функцията, exit може да се извика на всяко място в кода (което не е много добър подход).
- Добър подход за обработка на грешки е да се връщат с
  - „return <код или обект за грешка>“
  - още по-добре с „throw <обект за изключение>“.

И в двата случая грешката се обработва от извикващия метод и може да се предаде нагоре в йерархията на извикванията. Не се прескача йерархията на извикванията.

Частичен пример за exit()

```
int function1() {
   int retValue;
   ...
   if (...) { // грешка
      exit(1);
      // по-добре е грешката да се връща с return (не винаги е
   удачно) или още по-добре с throw и да се обработва от извикващата
   функция
   }
   ...

   return retValue;
}
```